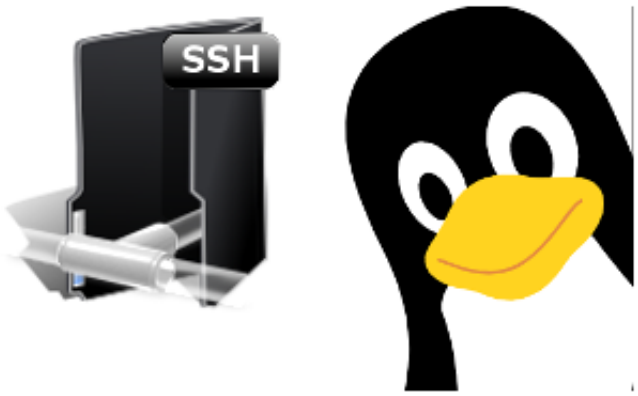
How to Copy large data directories between 2 Linux / Unix servers without direct ssh / ftp access between server1 and server2 other by using SSH, TAR and Unix pipes

Author : admin



In a Web application data migration project, I've come across a situation where I have to **copy / transfer 500 Gigabytes of data from *Linux server 1 (host A)* to *Linux server 2 (host B)***. However the **two machines doesn't have direct access to each other (via port 22) for security reasons** and hence I cannot use sshfs to mount remotely host dir via ssh and copy files like local ones.

As this is a data migration project its however necessery to migrate the data finding a way ... Normal way companies do it is to copy the data to ***External Hard disk storage*** and send it via some ***Country Post services*** or some employee being send in Data center to attach the SAN to **new server where data is being migrated** However in my case this was not possible so I had to do it different.

I have access to both servers as they're situated in the same Corporate DMZ network and I can thus access both **UNIX** machines via SSH.

Thanksfully there is a **small SSH protocol + TAR archiver and default UNIX pipe's capabilities hack that makes possible to transfer easy multiple (large) files and directories**. The only requirement to use this nice trick is to have SSH client installed on the *middle host* from which you can access via SSH protocol Server1 (from where data is migrated) and **Server2** (where data will be migrated).

If the hopping / jump server from which you're allowed to have access to Linux servers **Server1** and **Server2** is not Linux and you're missing the SSH client and don't have access on Win host to install anything on it just use portable mobaxterm (as it have Cygwin SSH client embedded)

Here is how:

```
jump-host:~$ ssh server1 "tar czf - /somedir/" | pv | ssh server2 "cd /somedir/; tar xf
```

As you can see from above command line example an SSH is made to ***server1*** a tar is used to archive the directory / directories containing my hundred of gigabytes and then this is passed to another opened ssh session to server 2 via *UNIX Pipe mechanism* and then *TAR archiver* is used second time to unarchive previously passed archived content. **pv** command which is in the middle is not obligitory though it is a nice way to monitor status about data transfer like below:

```
 500GB 0:00:01 [10,5MB/s]
[==================================================>] 27%
```

P.S. If you don't have **PV** installed install it either with apt-get on Debian:

```
debian:~# apt-get install --yes pv
```

Or on CentOS / Fedora / RHEL etc.

[root@centos ~]# **yum -y install pv**

Below is a small chunk of PV manual to give you better idea of what it does:

NAME
***pv* - monitor the progress of data through a pipe**

SYNOPSIS
pv [OPTION] [FILE]...
pv [-h|-V]

DESCRIPTION
pv allows a user to see the progress of data through a pipeline, by giving information such as time elapsed, percentage completed (with progress bar), current throughput rate, total data transferred, and ETA.

To use it, insert it in a pipeline between two processes, with the appropriate options. Its standard input will be passed through to its standard output and progress will be shown on standard error.

pv will copy each supplied FILE in turn to standard output (- means standard input), or if no FILEs are specified just standard input is copied. This is the same behaviour as cat(1).

A simple example to watch how quickly a file is transferred using nc(1):

pv file | nc -w 1 somewhere.com 3000

A similar example, transferring a file from another process and passing the expected size to pv:

cat file | pv -s 12345 | nc -w 1 somewhere.com 3000

Note that with too big file transfers using PV will delay data transfer because everything will have to pass

through another 2 pipes, however for file transfers up to few gigabytes its really nice to include it.

If you only need to transfer huge .tar.gz archive and you don't bother about traffic security (i.e. don't care whether transferred traffic is going through encrypted SSH tunnel and don't want to put an overhead to both systems for encrypting the data and you have some unfiltered ports between host 1 and host 2 you can run **netcat** on host 2 to listen for connections and forward .tar.gz content via netcat's port like so:

```
linux2:~$ nc -l -p 12345 > /path/destinationfile
linux2:~$ cat /path/sourcfile | nc desti.nation.ip.address 12345
```

Another way *to transfer large data without having connection with server1 and server2 but having connection to a third host PC* is to use **rsync** and good old SSH Tunneling, like so:

```
jump-host:~$ ssh -R 2200:Linux-server1:22 root@Linux-server2 "rsync -e 'ssh -p 2200' --stats --progress -vaz /directory/to/copy root@localhost:/copy/destination/dir"
```